仕事の進め方

はじめに

どういうふうに仕事してるかなと考えた時に
意識している部分について書いていく

目次

押さえておきたいこと

- 全体像を把握する
- 目標の深掘をする
- 作業の筋道をイメージする
- わからない部分があったら質問する
- 進捗はこまめに報告する

できたら嬉しいこと

- なるべくシンプルに考える
- DIとか疎結合を意識する
- 読みやすい文章を心がける
- 色々疑ってみる（建設的に）

押さえておきたいこと

全体像を把握する

どういう感じで業務、システムが動いているかをざっくり把握する
以下の粒度で理解していきたい（下に行くほど細かい）

- 何ができるものなのか
- 業務、操作フロー
- システム構成
- データフロー
- インターフェース（I/O）
    - API
    - プログラム

目標の深掘をする

操作手順だったり、最終的な出力について具体的なイメージ
を持って議論する

- ここでモックだったり、サンプルを作っておくといい
    - モック、APIの出力などは作成が簡単
    - 認識合わせの戻りのコストを抑えることができる
    - コスパ高い印象

最初の仮説は結構大雑把だったりするので、
Why（なぜ）をヒアリングして「顧客が本当に欲しいもの」
がなにかを認識のすり合わせを入念に行うべき

作業の筋道をイメージする

- 内部の細かい処理はさておき、インプット、アウトプットがどのようになるのか
  必要条件などをまとめる
- そのあと、どのような操作を行えば、期待した結果を出力できるかをイメージする
  (ぐるぐる回して、この条件の場合にフィルターして、こういうデータを導いて出力してみたいな感じで)
- どうすればいいのかわからない場合、空の `Class`, `Interface` を作って、
  コメントで動作を記載していく
  - 大まかに出してから、プログラムの処理の単位に分割していく
  - その後、デバッグだったり、テストコードでダミーデータを作り
    どのようにデータが作成されているか、確認しながら作成していく

わからない部分があったら質問する

前述の筋道すらつけられないような状態であれば、
現在の自分が持っている情報と仮説を話して認識が合っているのか確認したほうがいい

あとは、作業内容が全然頓珍漢なことをやっていて、
1日無駄にするとかも勿体無いので、筋道を立てた時点で確認をすべきかなと思います

よく言われる、 n 分考えてわからなかったら相談するってやつですね

進捗はこまめに報告する

ここの前述のわからない部分の話と類似するが、こまめに報告しておいたほうが、
作業の乖離を防げるのでやったほうがいいです

1日遅れたことを 1日遅れた後にわかるのと、
1時間遅れたことを 1時間遅れた後にわかるのは進捗の管理側としては
後者のほうが圧倒的に嬉しいので

できたら嬉しいこと

プログラミングの話もあります

必須ではないけど、ここまでできていると嬉しいなを書きました

なるべくシンプルに考える

色々なことを考慮する必要が出てきた場合、本当に複雑になることは少ない

Divide and conquer（分割して統治せよ）

上記の一言にすぎるが、複雑に見えるものは分割していくと
シンプルになったりすることが多い

関数を分割したり、クラスを適切に分割したり、やり方は色々あるので
関心の対象を分離、分割できるようにしましょう

DIとか疎結合を意識する

Dependency Injection (DI)、いわゆる依存性の注入はいろいろなものを差し替えることができるのでテストがしやすいし、結合度が減るので意識しましょう

作成の際のポイントとして、
外部インターフェースが絡む部分はすべて `constructor` から受け取るようにする
変換する処理なども `constructor` から受け取るようにすると他の場所でも使えるので考慮

疎結合は言わずもがな、全部がくっついていると
すべてを検証する必要が出てくるのでテストが大変になる。

なるべくAPIを分けたり、処理や画面、関数、クラスを分けたりして
1個の処理の単位を小さくして処理を完結させるようにすることを意識すること

読みやすい文章を心がける

例えば、以下の文章があったとする

> 当社の最新の製品は革新的なテクノロジーを駆使しており独自の特許を取得した物であり他社製品とは一線を画している為顧客はより高度なパフォーマンスと優れた利便性を体感できるでしょう。

句読点が打たれてなかったり、改行がないので読みにくい
また、すべてが一文で繋がっていたり、漢字が多く読みづらい

句読点を適切に区切って、改行も適切に加える
漢字も一部をかな文字に直すと、だいぶ読みやすくなる

> 当社の最新の製品は、革新的なテクノロジーを駆使しています。
> このテクノロジーは、独自の特許を取得したものであり、
> 他社製品とは一線を画しています。
> その結果、顧客はより高度なパフォーマンスと優れた利便性を体感できるでしょう。

漢字に関しては同音異義語になるものは漢字のほうがいいかなと思います
このあたりは感覚の問題なので、自分が読みやすいなと思う形に添削すればよいです

冗長な表現をなくして、本質的な部分だけを伝えるようにする

> 当社の最新製品は、独自のテクノロジーを活かしています。
> これにより、高度なパフォーマンスと利便性が実現され、
> 顧客に優れた体験を提供します。

なるべく視界に入った際、適切に文章が区切られているかを意識するとよいと思います。

色々疑ってみる（建設的に）

なんとなくふわっとみんながそう言っているから、で結論が決まりそうだったりした場合、
問題なさそうでも、対案を提示した方が良かったりすることが多い

対案が採択されることを目的としていない（採用されるときもあるが）
対案によって考慮が必要な点が洗い出されたりするのでオススメ

プログラミングに関しては
データパターンだったり、パラメータはどのような状態がありえるのか、
セキュリティ的に問題ないか、性能的に問題ないかなど、いろいろな視点で確認

批判だけを目的に否定するだけみたいなのはNG（言い方、気をつけよう）
設計やプログラミングは偉いからといって、
なにかが正しくなるとかはないので、安心して議論しましょう。
（間違えていたら、素直に「ごめんなさい」すればいいだけなので）

終わりに

まだいっぱいあるような気はするのですが、これくらいで

自分で言語化して発見もあったのでよかったです

参考になったら幸いです

ご清聴ありがとうございました